Really Paying Attention: A BERT+BiDAF Ensemble Model for Question-Answering

Andrew Ying
Department of Computer Science
Stanford University
Stanford, CA 94305
partying@stanford.edu

Abstract

Question answering is one of the most elusive disciplines within the realm of computer science and natural language processing. The arrival of Bidirectional Encoder Representations From Transformers (BERT) last year took the NLP world by storm. BERT's successes raised questions about whether pre-trained contextual embedding (PCE) models are poised to supplant the past era of designing and iterating on diverse, complex architectures for specific tasks. In this project, I sought to shed light on this question. First, I experimented with hyperparameter fine-tuning to better understand how to optimize BERT performance on SQuAD 2.0. Second, I explored the relationship between a baseline Bidirectional Attention Flow (BiDAF) model and a BERT model. Specifically, I created a BERT+BiDAF ensemble using a stochastic approach. The results were quite interesting: though the ensemble model appeared to decrease scores on the dev set, it ultimately performed better than standalone BERT on the test set, achieving a final F1/EM score of 77.165/73.609. Ultimately, however, the increments afforded by such an ensemble model were comparatively less than those realized through increasing BERT compute power via hyperparameter tuning, suggesting that we may indeed be in a "new era" of NLP dominated by pre-contextualized embeddings.

1 Introduction

Question answering (QA) is one of the most elusive disciplines within the realm of computer science and natural language processing. Often viewed as a litmus test of artificial intelligence approaching human-level abilities, QA involves processing a corpus of information to construct answers. In a world increasingly being changed by artificial intelligence, QA has proved its importance and significance by helping augment human cognition through tools such as Wolfram Alpha.

Yet, QA has proved difficult for a variety of reasons. First, QA algorithms are often reliant on having corpora that contain verbatim the actual answer. As a result, an often critical component of any QA model is selecting documents which might contain the answer. To abstract away this challenge and instead focus on locating the answer when it does exist, many QA datasets, such as the Stanford Question Answer Dataset (SQuAD), require that the answer to a given question is a span, or exact segment, from the inputed reading passage [8]. Additionally, QA algorithms are susceptible to adversarial examples, which can manifest themselves in a variety of ways. For example, a span of the inputed corpus could have a high exact match with the question but be about an entirely different subject, thus tricking the QA algorithm into picking the wrong answer. Finally, even with more recent versions of QA datasets such as SQuAD, QA algorithms have still been criticized for being unable to capture the nuances of human common sense, with notable examples being question answering in context (QuAC) and answering questions that require multiple steps of reasoning (HotPotQA) [2, 13].

NLP scholars have worked on reading comprehension for decades, tracing back to the Yale AI Project led by Roger Schank [9]. Yet major progress was not made until the last few years, with the release of large datasets which facilitated the training of deep neural network approaches. With researchers constantly testing new architectures, these neural network approaches, such as QANet, Bidirectional Attention Flow (BiDAF), and Attention-over-Attention, were as diverse as they were complex. Yet significantly, within the last year, the discipline of QA has experienced momentous breakthroughs with the release of Pretrained Contextual Embedding (PCE)-based approaches. Compared to models based on classic static word embeddings, models based on PCE incorporate context in order to capture the polysemous qualities of words, among many other benefits. Interestingly, these PCE models often required only one additional layer to adapt to various NLP tasks and achieve impressive performance.

This project seeks to answer the question of whether PCE models are poised to supplant the past era of designing and iterating on diverse, complex architectures. To do this, I experimented with hyperparameter fine-tuning to better understand how to optimize BERT performance on SQuAD 2.0. Concretely, I modified a PyTorch implementation of BERT from huggingface [6] to support the loading of models that have already been fine-tuned in order to facilitate the process of tuning hyperparameters to better understand what works best to improve PCE models. Second, I explored the relationship between a baseline BiDAF model and the BERT model. To do this, I wrote code to create an ensemble model that fuses the BiDAF baseline and BERT model. The motivation behind creating this ensemble model is that multiple learning algorithms can often achieve better predictive performances than could be obtained from any of the constituent models alone [7]. In the context of reading comprehension, a model like BiDAF which is specifically tailored to question answering might be able to rectify mistakes from the BERT for SQuAD model, which, as explained below, has a comparatively simple approach in terms of fine-tuning for the task of question-answering.

2 Related Work

2.1 Bidirectional Attention Flow for Machine Comprehension

The BiDAF Network was introduced at ICLR 2017. At the time, it achieved state-of-the-art results on SQuAD and the CNN/DailyMail cloze test. The modifications to the BiDAF Network that are unique to the Default Final Project are discussed below in the Approach section; for more information about BiDAF writ large, one can refer to the official whitepaper [10].

2.2 How BERT is Powered: The Transformer

At the forefront of related works is the monumental 2017 paper that introduced the Transformer architecture: "Attention Is All You Need" by Ashish Vaswani et al. from Google AI [11]. At the most basic level, the Transformer has encoder and decoder portions, as depicted in Figure 1.

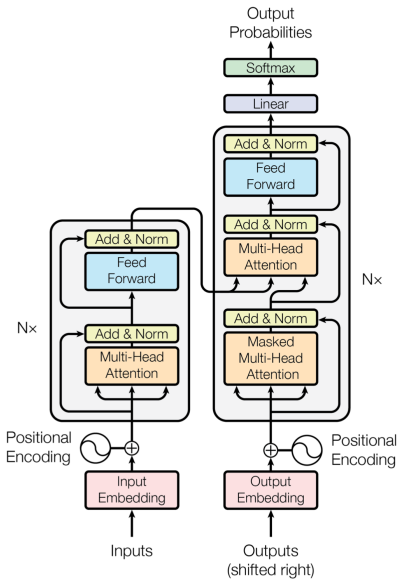
For language modeling purposes such as BERT, only the Encoder portion of Transformer is used. As displayed above, each encoder (on the left) has two sub-layers: a self-attention layer and a feed-forward neural network layer. The self-attention layer receives embeddings that are concatenated with positional encoding vectors in order to account for the order of inputed words. In contrast, the feed-forward layer is agnostic to dependencies between words, which means that operations can be executed in parallel in the feed-forward layer, leading to the impressive performance speedups associated with the Transformer [1].

Analogous to how hidden states in recurrent neural networks allow some degree of storage of previous words, self-attention is how the Transformer prices in understanding of contextual words while processing the current word of interest. However, Transformers directly model relationships between all words in a sentence, regardless of respective position [4]. Critically, this unique aspect of the Transformer architecture facilitates the Masked Language Modeling tasks that have proven effective in training BERT, extrapolated below in the Approach section.

2.3 Other Approaches: QANet

QANet combines local convolution with global self-attention for the specific purpose of QA. By abandoning RNN architectures, QANet achieved up to 13x faster speeds in training up to 9x faster

Figure 1: Internal structure of the Transformer [11].



speeds in inference. A key difference between QANet Encoder Blocks and Transformer Encoders is that QANet Encoder Blocks adopt character-level convolutions with max-pooling.

Like the BiDAF baseline, QANet is another brainchild of the era of devising and iterating on diverse, complex architectures to solve QA. Ultimately, this paper seeks to understand whether contextual embeddings pre-trained on massive corpora such as BERT are increasingly supplanting this era of designing architectures, or whether the relationship is more of a symbiotic one, *i.e,*, performance can be improved via ensembling.

2.4 BERT vs. Other Approaches in QA and Beyond

Qualitatively, BERT's pre-trained contextual embedding approach (as detailed above) differs with the feature-based training approach that characterizes models like ELMo. Noting the weaknesses of a one-size-fits-all word embedding in models such as GloVE, ELMo looks at the entire sentence before assigning each word in it an embedding. It uses a bi-directional LSTM trained on a specific task to be able to create those embeddings.

Quantitatively, when first released, On SQuAD v1.1, BERT achieved 93.2% F1 score (a measure of accuracy), surpassing the previous state-of-the-art score of 91.6% and human-level score of 91.2%. Additionally, below is a numerical figure that quantifies BERT's performance compared to other state-of-the-art models such as ELMo on the GLUE benchmark, which tests language models on a set of 9 diverse Natural Language Understanding (NLU) tasks. As depicted, BERT improved on the state-of-the-art performance by 7.6%.

Figure 2: BERT versus other SOTA models on the GLUE benchmark [4].

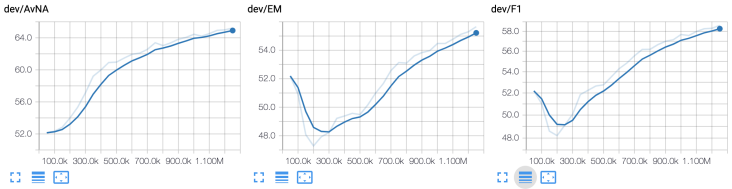
Rank	Model	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	QNLI	RTE
1	BERT: 24-layers, 1024-hidden, 16-heads	80.4	60.5	94.9	85.4/89.3	87.6/86.5	89.3/72.1	86.7	91.1	70.1
2	Singletask Pretrain Transformer	72.8	45.4	91.3	75.7/82.3	82.0/80.0	88.5/70.3	82.1	88.1	56.0
3	BiLSTM+ELMo+Attn	70.5	36.0	90.4	77.9/84.9	75.1/73.3	84.7/64.8	76.4	79.9	56.8

3 Approach

3.1 The Baseline: BiDAF Model

The baseline model is based on BiDAF. The embedding layer is at a word-level rather than a character-level, projecting embeddings to have dimensionality H and applying a Highway Network to refine the embedded representation. In the encoding layer, a bidirectional LSTM helps the model incorporate temporal dependencies between timesteps of the embedding layer's output. The unique part of the attention layer is that attention flows both ways, from the context to the question (C2Q) and the question to the context (Q2C). In the Modeling Layer, a bidirectional LSTM refines the output of the attention layer before the Output Layer performs softmaxes to produce vectors of probabilities p_{start} and p_{end} to get answers to the question[1].

Figure 3: Sample TensorBoard visualization of BiDAF Baseline Model Results after applying Adadelta optimization.



3.2 Breaking Down BERT

3.2.1 Overview

BERT produces a state-of-the-art language model. At the highest level, its architecture is a trained stack of Transformer Encoders as described in the Related Work section [4].

To tailor Transformers to language modeling, BERT has a unique approach to generating representations for each input. Specifically, BERT input embeddings are the sum of the token embeddings and their corresponding segment and position embeddings. The token embeddings are based on the WordPiece model [12], which is often thought of as a median between character-based and word-based models that effectively captures and exploits linguistic combining forms such as prefixes and suffixes. The segment and position embeddings are used for BERT pre-training and are detailed further in the following section.

3.2.2 Transformers for Language Modeling: BERT, Masked LM (MLM), and Next Sentence Prediction (NSP)

One of the most notable breakthroughs of the BERT whitepaper centers around its twofold approach to training the Transformer Encoder Stack. Training is performed in two main ways: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). Both of these methods facilitate BERT being the first deeply bidirectional, unsupervised language representation.

In MLM, as depicted on the bottom layer of Figure 2, 15% of the words in sequences inputed to the BERT substructures are first replaced with a [MASK] token. The model then is charged with predicting vectors for the masked words bidirectionally. The act of randomly deleting words is significant because it circumvents the issue of words indirectly "seeing itself" in a multilayer model.

Another important part of BERT training is Next Sentence Prediction (NSP), wherein the model learns to predict whether a secondary sentence follows from a first. NSP is facilitated by the `[SEP]` tokens, which delineate where the first sentence ends and the second one begins. Then, a classifier layer, consisting of a feed-forward neural network and a softmax, is added on top of the Transformer Encoder Stack. The deeper idea behind NSP is that the model learns to phrases might be related to each other, setting up potential applications to NLP tasks such as QA.

[1]There are many equations left out here (such as calculation of similarity matrix for attention layer) due to space limitations; for more information on the Baseline, including Training Details (e.g. the Loss Function) and how to predict no-answer, please refer to the DFP handout.

Figure 4: Masked Language Modeling (MLM) [5].

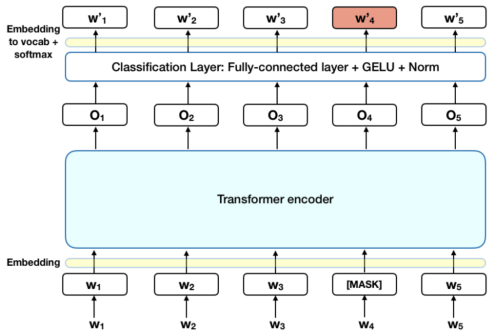
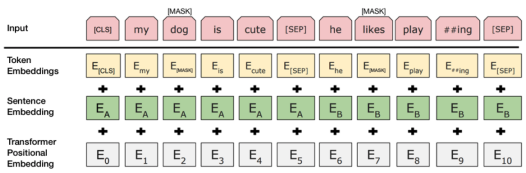


Figure 5: BERT Sentence Pair Encoding [4].

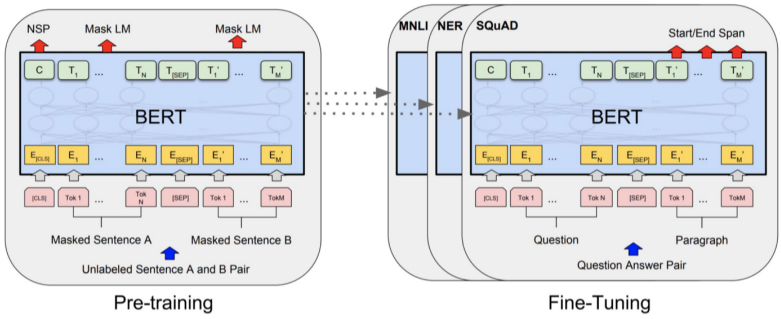


When training BERT, Masked LM and Next Sentence Prediction are trained together, with the idea that minimizing the combined loss function of MLM and NSP will yield the best representations.

3.2.3 Fine-Tuning BERT

BERT's simplicity is especially apparent when it comes to fine-tuning BERT to other tasks such as name-entity recognition, natural language inference, and question answering. For all these tasks, one can simply learn an additional layer built on top of the core model, as depicted below in Figure 5.

Figure 6: General method of fine-tuning BERT [4].



In the context of QA, the key method of tailoring BERT to SQuAD is learning two extra vectors $S \in \mathbb{R}^H$ and $E \in \mathbb{R}^H$ (where H is the size of the hidden layer) that denote the beginning and end of the answer, respectively. Formally, the equation that describes the probability that a word i is the beginning of an answer span is:

$$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}} \tag{1}$$

S is replaced with E to calculate the word with the maximum probability of being the end of an answer span. The loss function is then the log likelihood of the correct start and end positions [4].

3.3 Ensembling BERT and BiDAF

I wrote an original script to ensemble the BERT and BiDAF models (`ensemble.py`). Specifically, the script takes the .csv prediction files from the BiDAF and BERT models and ensembles them using a stochastic approach.

I settled on a randomized model where when there is disagreement, between the predictions, BERT prevails 0.98 of the time and BiDAF prevails 0.02 of the time. The motivation behind it is that there are cases, however improbable (see section 5.3), where BiDAF does outperform BERT, and it is desirable to squeeze out those performance gains possible.

4 Experiments

4.1 Data and Evaluation Metric

The datasets being used are provided by SQuAD with some modification from the Stanford CS224N Teaching Team. Evaluation Metrics are EM and F1. Details can be found in the DFP Handout.

4.2 Experimental Details

There were two main experiments run for this project:

1. **Tuning the HuggingFace PyTorch BERT model hyperparameters to increase performance.**

   The two finetuned BERT models are denoted BERT-1 and BERT-2 (trained on NV6 machine courtesy of Microsoft Azure), with specific hyperparameter details below:

Model	BERT-1	BERT-2
Train Batch Size	2	8
Learning Rate	3e-5	3e-5
Train Epochs	1.0	3.0
Max Seq Length	384	384
Doc Stride	128	128
16-Bit Float Precision	True	True

2. **Ensembling BERT and BiDAF to increase performance.**

   As detailed above, the ensembling heuristic I settled on was a stochastic model where when there is disagreement, BERT prevails 0.98 of the time and BiDAF prevails 0.02 of the time. The ensembling script was run after the BiDAF baseline and BERT models made their predictions. Implications of this choice are displayed in the Results section and discussed in the Analysis section.

4.3 Results

The main results of the experiments are summarized in the table below. Note that the first two BERT+BiDAF ensemble models do not have Test EM/F1 scores because they demonstrated diminishing performance on the dev sets. Due to the 3-run constraint on the test leaderboard, only the last ensemble model was chosen to submit to the test leaderboard.

Model Description	Dev EM	Dev F1	Test EM	Test F1
BiDAF Baseline	55	58	56.298	59.920
BERT-1	72.590	75.394	72.354	74.903
BERT-2	74.136	77.450	73.542	76.848
BERT+BiDAF (0.25 threshold)	69.148	72.365	N/A	N/A
BERT+BiDAF (0.10 threshold)	73.097	76.318	N/A	N/A
BERT+BiDAF (0.02 threshold)	73.786	77.074	73.609	77.165

5 Analysis

5.1 Analysis of Experiment 1: BERT Hyperparameter Tuning

The hyperparameter tuning section of my project was quite straightforward. I started with a smaller fine-tuning batch size of 2 (EM/F1 72.590/75.394), but increasing the size of the batch to 8 yielded stronger results, increasing the Dev EM and F1 scores by around 2 points (EM/F1 74.136/77.450). This makes sense because more tokens lead to increased granularity in fine-tuning. It is worth noting that increasing the batch size past 8 was infeasible on the NV6 machine due to memory constraints. However, the default train batch size is 32, and according to the SQuAD 2.0 leaderboard, the raw BERT single model with 32 train batch size achieves EM/F1 of 80.005 and 83.061. Such is the difference that compute power makes, and this will be further discussed below when assessing and quantifying the impact of ensembling.

5.2 Analysis of Experiment 2: Ensembling

There is much to be said and analyzed about the stochastic approach to ensembling described in the Approaches section. Admittedly, the approach is more of a makeshift method. However, I viewed it as an approximation of the fact that occasionally, the BiDAF model does yield a better answer than the BERT model. Indeed, as demonstrated in (E3) in the following section, there are indeed cases, however infrequent, where BERT's bells and whistles from massive pre-training, do err, and we might be able to randomly rectify those mistakes by ensembling with BiDAF.

Randomness is exactly what manifested in final results. Allowing BERT to dominate 0.75 of the time when there was a disagreement saw a decrease of around 5/4 points in EM/FM from the standalone BERT-2 model, and decreasing the threshold to 0.02 seemed to improve the scores back towards that of the standalone BERT-2 model. However, on the test set, the ensembling script managed to improve performance by 0.6 EM points and 0.3 F1 points.

Though a stochastic approach means we cannot extract the most meaning from this improvement, the improvement does indicate that the stochastic approach is able to exploit where BiDAF may rectify some of BERT's errors. Indeed, the following section of the paper contains examples where BiDAF outperforms BERT and discussion about possible reasons why.

Ultimately, a large reason why ensembling did not produce monumental gains in the context of this project is because ensembling tends to yield the greatest impact when there exists great diversity between the models. Both BERT and BiDAF put a large emphasis on attention in their architectures; perhaps the lack of diversity is the reason why most ensembled models on the current leaderboard do not involve BiDAF in any capacity. Based on the results of this project, however, it does seem that the upside of tuning hyperparameters and increasing compute power has much larger upside (≈ 10 times more gain in score) than ensembling. The broader implications of this difference will be discussed further in the conclusion.

5.3 Commentary on Selected Examples

Below are some examples that highlight the differences between BERT and BiDAF.

Which courts have a duty to interpret domestic law as far as possible? (E1)

Here, the relevant context is "Fourth, national courts have a duty to interpret domestic law as far as possible." For this example, the BERT model correctly predicted "national courts," but the BiDAF Model predicts "Fourth, national courts." This is a nice demonstration of the power of BERT pre-training; because the BERT Encoding Blocks are pre-trained on large corpora, the dot product equation (1) under the "Fine-tuning BERT" section will likely give a low probability to "Fourth" being a valid start vector because "Fourth" is unlikely to ever be associated with "national" or "courts" in pre-training. However, the BiDAF model would be oblivious to this issue, as it does not have the luxury of extensive pre-training, so it simply applies attention to the beginning of the sentence and includes "Fourth."

What Yuan policies did Muslims like? (E2)

Here, the relevant context is "Some policies of the Yuan Emperors severely discriminated against Muslims, restricting Islamic practices like circumcision." For this example, the BERT model correctly predicted "No Answer" but the BiDAF Model predicts "circumcision." Once again, the pre-training of BERT displays its power in handling the polysemous word "like." The BiDAF model, using GloVe vectors, is weaker on this issue, likely picking the wrong answer because "Islamic" and "Muslim" are close in high-dimensional space and the word "like" is attended to in both C2Q and Q2C.

What happens when all parts of a plant become infected? (E3)

Here, the relevant context is "When a part of a plant becomes infected, the plant produces a localized hypersensitive response." Interestingly, here the BiDAF model correctly predicts "No Answer" while the BERT model predicts "hypersensitive response." There does not to be a clear-cut answer to why this discrepancy happens; perhaps the BERT threshold necessary to generate "No Answer" needs to be more sensitive (indeed, the choice for all BERT models was -1 for the `diff_threshold`.) In comparison, the BiDAF model's GloVe vectors can quickly come to a consensus that "all" and "part" are very different such that $p_{\text{start}}(0)$ and $p_{\text{end}}(0)$ are greater than the corresponding values for "hypersensitive response." In this sense, perhaps the BiDAF model's adaptation to SQuAD 2.0 specifically is more robust.

6 Conclusion

This nexus question this project set out to answer was whether contextual embeddings pre-trained on massive corpora such as BERT are increasingly supplanting the past era of designing and iterating on complex architectures, or whether the relationship is more of a symbiotic one. The results of the experiments seem to support that researchers who lionize BERT as spearheading a new era in NLP may be correct, as the improvements afforded by increasing compute in ways such as increasing the batch size of BERT fine-tuning dwarf the small increments afforded by the ensemble model. Ultimately, it is important to note that even the examples above were hand-picked; there are far more examples where BiDAF is simply wrong, and the few instances where BiDAF beats BERT seem to be the exception rather than the norm, as evidenced by the ≈ 20 point difference in EM and F1 scores between BERT-2 and BiDAF.

If it is true that pre-trained contextualized embeddings are the future of NLP, however, there is a dark, or perhaps inconvenient, side to such a revolution. Indeed, the success of BERT is yet another case for those who claim that the way to solve most problems in AI is to "throw more compute at it." It is worth noting that BERT's reliance on compute power does lead to one of its most important weaknesses, namely, that its bidirectional approach (MLM) converges much slower than left-to-right approaches.

Certainly, there are considerable limitations to my work. Most notably, ensembling has previously been done in many ways that are much more precise than the stochastic approach pursued in this project. An area for future work is thus developing a more sophisticated ensembling heuristic, such as learning some weighting over the predictions (i.e. one more linear layer) and seeing if the weighted combination makes any difference.

In terms of other avenues for future work, with more time and collaborators, I would have liked to explore how to more deeply and comprehensively integrate the BERT and BiDAF models to target scenarios just like (E3) in the Analysis section above. Whereas my approach was more end-facing, taking the predictions that each model created and ensembling them stochastically, I feel that there could be positive results from replacing the embedding layer of the BiDAF models with BERT to more deeply contribute to answering the question of whether we are in a new PCE-dominated era of NLP. Along these lines, with sufficient credits, I would have liked to really play around with all the hyperparameters in section 4.2, including maximizing number of train epochs and maximum sequence length, to see if compute power is really all one needs. In a different direction, I would have liked to implement part of the Attention over Attention paper [3], which resonates with me as I liken AoA's effectiveness with human metacognition during reading comprehension.

References

[1] Jay Alammar. *The Illustrated Transformer*. June 2018. URL: http://jalammar.github.io/illustrated-transformer/.

[2] Eunsol Choi et al. "QuAC : Question Answering in Context". In: *CoRR* abs/1808.07036 (2018). arXiv: 1808.07036. URL: http://arxiv.org/abs/1808.07036.

[3] Yiming Cui et al. "Attention-over-Attention Neural Networks for Reading Comprehension". In: *CoRR* abs/1607.04423 (2016). arXiv: 1607.04423. URL: http://arxiv.org/abs/1607.04423.

[4] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[5] Rani Horev. *BERT Explained: State of the art language model for NLP*. Nov. 2018. URL: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[6] Hugging Face Inc. *PyTorch Pretrained BERT: The Big & Extending Repository of pretrained Transformers*. https://github.com/huggingface/pytorch-pretrained-BERT/. 2018.

[7] Richard Maclin and David W. Opitz. "Popular Ensemble Methods: An Empirical Study". In: *CoRR* abs/1106.0257 (2011). arXiv: 1106.0257. URL: http://arxiv.org/abs/1106.0257.

[8] Pranav Rajpurkar et al. "SQuAD: 100, 000+ Questions for Machine Comprehension of Text". In: *CoRR* abs/1606.05250 (2016). arXiv: 1606.05250. URL: http://arxiv.org/abs/1606.05250.

[9] Roger C. Schank and Robert P. Abelson. *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. Hillsdale, NJ: L. Erlbaum, 1977.

[10] Min Joon Seo et al. "Bidirectional Attention Flow for Machine Comprehension". In: *CoRR* abs/1611.01603 (2016). arXiv: 1611.01603. URL: http://arxiv.org/abs/1611.01603.

[11] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[12] Yonghui Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144 (2016). arXiv: 1609.08144. URL: http://arxiv.org/abs/1609.08144.

[13] Zhilin Yang et al. "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering". In: *CoRR* abs/1809.09600 (2018). arXiv: 1809.09600. URL: http://arxiv.org/abs/1809.09600.